Object Detection using Vision Transformer as a Backbone

Pragya Sharma
PhD Student, ECE
Duke University
Durham, NC 27708

Mansi Choudhary
PhD Student, ECE
Duke University
Durham, NC 27708

1. Introduction and Problem Statement

Computer vision plays a pivotal role in numerous applications, such as autonomous driving, healthcare, security surveillance, and augmented reality, among others. Its significance lies in its ability to enable machines to interpret and understand visual information, mimicking human vision to varying extents. Recent advances in computer vision have resulted in the development of sophisticated models that excel at interpreting complex visual data. Despite the impressive strides made in computer vision tasks made by the convolutional neural networks (CNNs) based models like Faster R-CNN [1], real-world images pose a significant challenge. Their inherent complexity and variability, including occlusion, diverse lighting, and object size/shape variations, expose the limitations of traditional CNN-based models.

The recent evolution of machine learning has resulted in significant developments in natural language processing, leading to the creation of the Transformer model, first proposed in [2]. This model, initially designed for sequence-to-sequence tasks in natural language processing, has demonstrated remarkable capabilities in capturing long-range dependencies and global context through its self-attention mechanism. Adapted into the Vision Transformer (ViT) [3] for computer vision tasks, the ViT has outperformed CNNs in image classification tasks. By dividing an image into fixed-size patches, linearly embedding them, and then processing them as a sequence, the ViT is able to capture the long-range dependencies and global context that CNNs often miss.

This project aims to explore how the ViT can be effectively utilized as a backbone architecture for object detection, comparing its performance and efficiency against traditional CNN-based models. The goal is to leverage the unique capabilities of self-attention in transformers by employing a pretrained ViT backbone to enhance performance of downstream vision tasks such as object detection, ultimately improving their performance and applicability in real-world scenarios.

2. Motivation

Vision Transformers (ViT) offer a compelling foundation for object detection, addressing challenges in conventional convolutional neural networks (CNNs). ViT's self-attention mechanism excels at capturing long-range dependencies, enabling a better understanding of image structure which is critical for precise object localization, classification, and recognizing object relationships. Another significant advantage of ViT over CNNs in object detection lies in its scalability, allowing flexible adjustments for diverse computational constraints and performance requirements. This scalability makes ViT a versatile tool that can be tailored to diverse applications, from resource-constrained mobile devices to high-performance computing systems. Furthermore, ViT's ability to generalize from limited data, thanks to pre-training on large-scale datasets, makes ViT promising for object detection in data-scarce domains. ViT's adaptability enhances robustness, addressing variations in object scale, occlusion, and background clutter. Leveraging ViT as a backbone aims to contribute to more efficient and effective object detection models, extending its success in image classification to diverse real-world scenarios.

Employing ViT as a backbone architecture may result in enhanced robustness of the model, allowing it to effectively handle variations in object scale, occlusion, and background clutter by selectively focusing on specific image regions. Moreover, given ViT's impressive track record in image classification tasks, it is plausible that these advantages could extend to other image tasks such as object detection. We believe that ViTs can be a unified backbone for diverse computer vision tasks such as image classification, object detection, gesture recognition etc, expanding its applicability to a plethora of real-world scenarios

Ultimately, our goal is to contribute valuable insights and practical guidelines to the field, paving the way for more efficient, versatile and robust models adaptable to a wide range of computer vision challenges that leverage the unique strengths of Vision Transformers.

3. Related Work

The landscape of computer vision tasks such as object detection is being reshaped by the emergence of Transformers, showcasing performance competitive with highly complex Convolutional Neural Networks (CNNs). Meta's DETR (DEtection TRansformer) [4] exemplifies this shift, offering a streamlined end-to-end pipeline that leverages

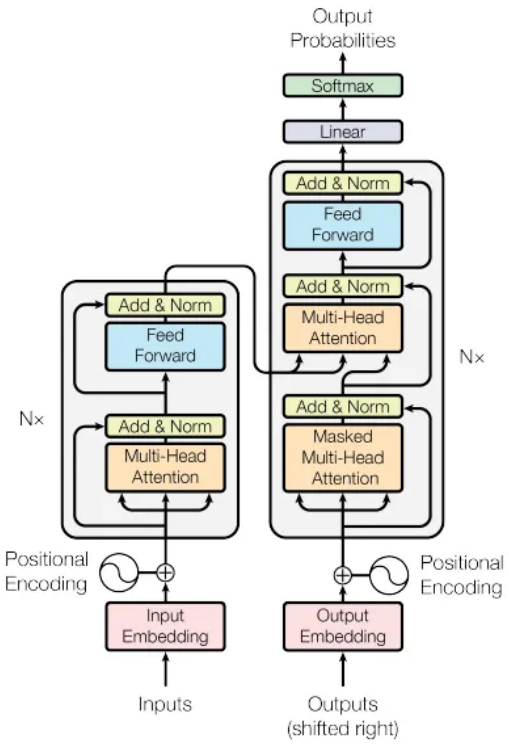

Figure 1. Transformer architecture as proposed in [2]

Transformers for object detection with a CNN backbone. DETR trains all model weights from scratch, limiting its flexibility. In contrast, our model adopts a more efficient approach. We leverage a pre-trained backbone to extract rich image features, significantly reducing training time and complexity. Notably, we only fine-tune a single layer on top of the backbone for the specific object detection task. This targeted approach maintains excellent performance while preserving the backbone's generalizability to other tasks.

While other models like SWIN [5] and ViTDet [6] also utilize pre-trained backbones, they rely on complex Feature Pyramid Networks (FPNs) - either plain or hierarchical - to aggregate multi-scale features. These networks, while effective, add significant complexity and limit downstream transferability. Our model stands out for its simplicity and modularity. By focusing on fine-tuning a single layer, we achieve competitive performance with minimal training overhead. Additionally, our architecture readily extends to other downstream tasks by simply adding additional heads, without incurring the complexity burden of FPNs.

4. Implementation Methodology

In the rapidly evolving field of computer vision, the advent of Vision Transformers (ViT) has marked a significant shift from traditional convolutional neural networks. This report delves into the implementation of an advanced object detection model utilizing a ViT backbone, specifically the 'google/vit-base-patch16-224' configuration. Our model leverages the potent feature extraction capabilities of ViT, integrating it with a custom object detection head for precise localization and classification of objects within images. The methodology is applied to the comprehensive COCO dataset, a benchmark in the field for object detection tasks. This implementation not only demonstrates the versatility and efficacy of ViT in handling complex image-based tasks but also serves as a testament to the model's adaptability in processing diverse and intricate datasets. The following sections will provide a detailed exposition of the model architecture, data preparation and preprocessing, training procedures, and evaluation metrics, offering a thorough insight into the nuances of deploying Vision Transformers for object detection.

4.1. Model Architecture

At the heart of our object detection model is the Vision Transformer (ViT), specifically the 'google/vit-base-patch16-224' [7] configuration. The reason we chose the ViT as our backbone stems from its exceptional ability to process images as sequences of patches, which allows for a global understanding of the image context. Unlike traditional convolutional networks that analyze only local features, the ViT is able to dissect images into fixed-size patches, encode them into linear embeddings, and then process these embeddings through a series of transformer blocks, each reinforced with layer normalization and multilayer perceptrons (MLPs). The layer normalization ensures stable and efficient training dynamics, while the MLPs, with their nonlinear capabilities, are instrumental in the model's proficiency in deciphering complex image features. The multi-head self-attention mechanism, in particular, allows the model to weigh and integrate information from different parts of the image, thereby capturing complex, inter-patch relationships and patterns. This approach enables to model to capture intricate patterns and relationships within the image, making it highly effective for feature extraction. In our implementation, the ViT backbone is responsible for generating rich feature representations from input images, which are then fed into the subsequent object detection head for further analysis.

Building upon the robust features that are extracted by the ViT backbone, our model incorporates a custom object detection head designed to perform two critical tasks: classifying objects and predicting their bounding boxes. This head comprises two primary components: a classifier and a box regressor. The classifier is a linear layer that maps the extracted features to class probabilities, predicting the likelihood to each object class within the image. The box regressor, on the other hand, also a linear layer, is tasked with determining the precise coordinates of the bounding boxes for each detected object. The bounding box predictions are made in the format of four coordinates, representing the corners of the box. This dual approach ensures that our model not only recognizes the presence of objects but also accurately pinpoints their locations, a critical aspect of effective object detection.

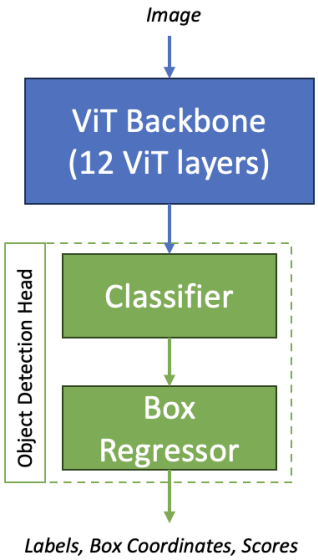

Figure 2. A simplified figure of the implemented Object Detection model

The seamless integration between the ViT backbone and the object detection head is a testament to the model's architectural efficiency. As an image enters the model, it is dissected into patches and transformed into a sequence of embeddings. These embeddings are then intricately processed through the self-attention layers of the ViT, enabling the model to understand the global context and intricate details of the image. The resulting feature vectors, rich in contextual information, are then channeled into the object detection head. Here, the classifier interprets these features to ascertain the likelihood of each object class, while the box regressor utilizes the same features to determine the exact bounding box coordinates for each detected object. This dual-pathway processing ensures that each aspect of object detection - classification and localization - is handled with precision and efficiency. The end-to-end architecture of our model not only leverages the advanced capabilities of Vision Transformers but also aligns them specifically to meet the demands of accurate and reliable object detection.

4.2. Data Preparation and Preprocessing

Our object detection model is trained on the COCO dataset [8], a benchmark in the field known for its diversity and complexity. This dataset comprises a vast array of images across various categories, making it ideal for training robust object detection models. The first step in data preparation involves parsing the COCO dataset annotations. This process is crucial as it involves extracting meaningful information from the dataset, such as object categories, bounding box coordinates, and image metadata. To achieve this, we developed a custom parser that efficiently processes the dataset's JSON formatted annotation files. This parser not only retrieves the necessary information but also organizes it in a structured manner, facilitating easy accessibility and manipulation during the training phase. The annotation data, along with the corresponding images, form the foundation upon which our model learns to detect and classify objects.

Preprocessing the images is a critical step in our model's training pipeline. Given the diverse nature of the COCO dataset, the images come in various sizes and resolutions. To maintain consistency and ensure efficient training, we first resize all images to a uniform dimension of 224x224 pixels. This resizing is accompanied by maintaining the aspect ratio to avoid distortions. Following resizing, we apply a series of transformations to the images. These include converting the images to tensors, which are then normalized using predetermined mean and standard deviation values. This normalization standardizes the pixel values across the dataset, aiding in faster and more stable model training. Additionally, data augmentation techniques such as random cropping, flipping, and rotation are employed to enhance the model's robustness and its ability to generalize across varied and unseen data.

The final aspect of data preparation involves the creation of target data for object detection. The target data comprises the bounding box coordinates and the associated class labels for each object in an image. In our implementation, bounding boxes are adjusted to align with the resized images. The coordinates are normalized relative to the image size, ensuring that the model learns to predict location and size regardless of the original image dimensions. For the class labels, we employ a mapping strategy to convert the categorical labels from the COCO dataset into numerical indices. This step is crucial for training the classifier in our object detection head, as it requires numerical inputs. Furthermore, we ensure that this mapping is consistent across the dataset to maintain label integrity. Together, the processed images and the meticulously prepared target data form a comprehensive dataset, primed for training our advanced object detection model.

4.3. Training Procedures

The training process commences with the careful initialization of our object detection model. Leveraging the pretrained 'google/vit-base-patch16-224' Vision Transformer as the backbone provides a substantial advantage, as it brings along an extensive understanding of visual features learned from a large-scale dataset like ImageNet [9]. This pretraining phase accelerates the model's ability to adapt to object detection tasks, as it builds upon already learned complex feature representations. Utilizing PyTorch's dynamic computation graph and GPU acceleration, the model efficiently handles the high-dimensional data involved. For batch processing, we utilize PyTorch's DataLoader with a specifically tuned batch size that optimizes the trade-off between computational efficiency and the model's capacity for learning from diverse data in each training iteration. This batch size is crucial for maintaining an efficient learning process, particularly given the extensive and varied nature of the COCO dataset.

Each training batch undergoes a forward pass through the model, yielding predictions for object classes and bounding box coordinates. Our loss function is meticulously designed to cater to the intricacies of object detection. It combines a Cross-Entropy Loss for class prediction accuracy and a Smooth L1 (Huber) Loss for bounding box precision. This combination is critical for ensuring the model effectively learns both aspects of object detection – identifying the correct object classes and accurately predicting their locations. The total loss is calibrated with a weighted sum approach, allowing us to fine-tune the model's sensitivity to classification versus localization accuracy. To enhance the model's learning capability, techniques like label smoothing for classification loss and IoU-based loss adjustments for bounding box predictions are also considered, providing a more nuanced and effective training regime.

The backpropagation algorithm is employed to calculate gradients, which are crucial for the model's learning. We utilize the Adam optimizer for its adaptive learning rate capabilities, which helps in navigating the complex parameter space effectively. To further refine the learning process, we incorporate learning rate scheduling, which methodically adjusts the learning rate, aiding in stabilizing and improving the model's convergence over epochs. Model training is punctuated by periodic validation checks, where the model's performance is evaluated against a separate validation dataset. This step is vital for ensuring that the model not only performs well on the training data but also generalizes effectively to new, unseen data. These checkpoints are not just fail-safes against potential interruptions but also strategic tools for model evaluation and selection, allowing us to restore the best-performing model based on validation metrics.

4.4. Model Inferencing and Visualization

During model inferencing, we perform a forward pass through the model and pass the model outputs through post processing steps to get the appropriate format for visualization and evaluation. In the forward pass, the model generates logits corresponding to each class for a hundred bounding boxes associated with each figure in the input data. These logits represent the model's confidence or probability for each class label within each bounding box. The label with the highest probability is selected as the assigned label for the respective bounding box, and the associated probability value becomes the confidence score for that box. To refine the results and focus on more confident predictions, a threshold value is set. We explore threshold values ranging from 0.5 to 0.75 and choose 0.65 as the optimum threshold for the validation data inferencing.

Bounding boxes associated with prediction scores falling below this determined threshold are systematically filtered out. Consequently, this curation process effectively trims down the number of bounding boxes, retaining only those with higher confidence levels.

Once this filtering step is completed, the remaining labelled bounding boxes can be visualized for each image. This visualization provides a representation of the model's predictions, highlighting the most probable object locations and their associated class labels.

5. Evaluation and Results

We utilized Meta's DETR (DEtection TRansformer) [] as the baseline for comparison, a transformer specifically trained for end-to-end object detection. DETR demonstrates accuracy and run-time performance on par with the well-established and highly-optimized Faster R-CNN baseline on the challenging COCO object detection dataset. (insert figure) We intended to use Average Precision as our metric of comparison. Average Precision provides a comprehensive measure of a model's precision across different levels of recall. We evaluated DETR and our model on the COCO val2017 object detection dataset. For the evaluation, we prepared a results file of the generated annotations in a specific data format:

```
[{
"image_id" : int,
"category_id" : int,
"bbox" : [x,y,width,height],
"score" : float,
}]
```

The model results compare it with a provided annotation file. The results of the DETR models are reported in figure 3.

The COCO evaluation results of our model showed poor performance, confirming our earlier observations through visual inspection. Among the tasks the model was trained to do (object detection, localization and classification), we think the model fails in classifying the object labels correctly, but was able to somewhat localize the boxed objects. While neither visual inspection nor COCO evaluation results met our expectations, the findings offer valuable lessons and inform our next steps for model development

6. Finetuning and Future Work

As mentioned above, our model performed poorly when it came to classifying the object's labels correctly. However, through visual inspection and manual comparisons of ground truth bounding boxes and predicted bounding boxes, we did notice that the model performed better when it came to localizing the objects. Because this was the case, we tried many different ways to increase model performance on the classification of objects.

The first of these ways was to train all the ViT Layers in the backbone, along with the embeddings, and not just the custom object detection model. This would allow for a more holistic and adaptable approach for feature extraction, enabling the model to fine-tune its visual representations. Theoretically, by doing so, the model should be able to better capture the intricate details of objects, thereby improving both localization accuracy and object classification.

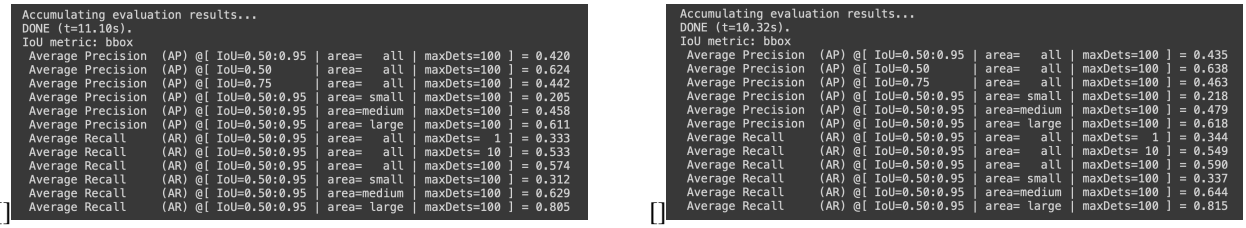

Figure 3. CocoEval results from (a) DETR-Resnet-50 and (b) DETR-Resnet-101 models from Meta AI

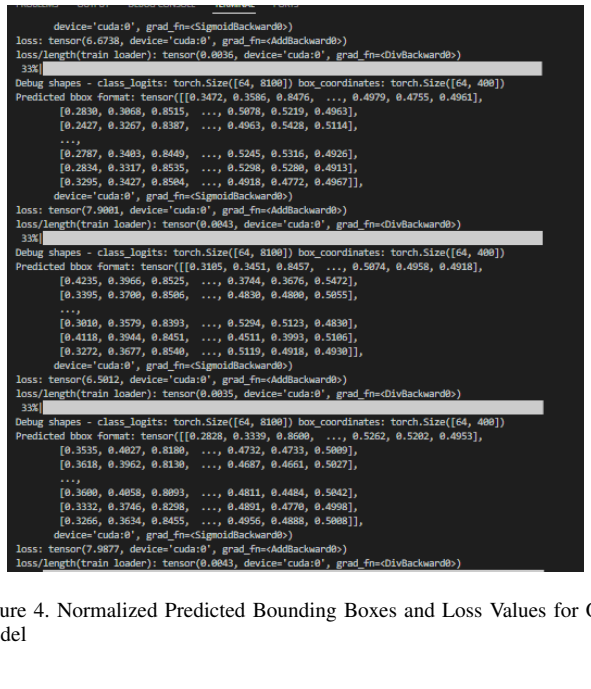

Figure 4. Normalized Predicted Bounding Boxes and Loss Values for Our Model

However, since this approach does require careful attention to hyperparameters and other variables during training, we were not successful when it came to object classification.

Our next step was to focus improving the model architecture, primarily, the layers that our object detection model consisted of. We believed that the poor classification performance could most likely have been because of the simplicity of the model. So, we went ahead and added batch normalization and a fully connected linear layer to our custom object detection head. This would allow or better feature extraction and finer-grained class predictions. The batch normalization helps stabilize training by normalizing the input to each layer, while the fully connected linear layer can capture more complex patterns and relationships among features, potentially leading to more accurate and robust object classifications. These architectural enhancements aimed to bridge the gap between object localization and classification, ultimately contributing to a more effective and powerful object detection model. Even with these changes, we noticed very little improvement in our model's object classification performance.

Moving on, we changed our optimizer for Adam to AdamW, which accounted for weight decay. This adjustment would allow our model to better manage regularization during training. AdamW's inclusion of weight decay helps prevent overfitting by encouraging the model to reduce the magnitude of its learned weights, promoting more generalizable feature representations. By incorporating AdamW into our training process, we aimed to strike a balance between effective learning and regularization, enhancing the model's capacity to learn discriminative features and improve its overall performance on our object detection task. Along the same lines, we made sure that our target bounding box coordinates were normalized, just like the predictions, to ensure consistent and meaningful comparisons during training. Normalizing both the target and predicted bounding box coordinates allowed us to mitigate issues related to scale and aspect ratio variations across different objects in our dataset. This normalization facilitated smoother convergence during training and made it easier for the model to learn the relationships between objects' locations and sizes, ultimately leading to more accurate and robust object localization, but still did not impact the performance of the model on object classification.

Since none of these methods seemed to be working for us, we decided to take a deeper look at the DETR model and its architecture, hoping to understand the differences that lie between our object detection approach and theirs. One of these differences was the fact that DETR has a convolutional neural network (CNN) as their backbone, whereas our model consists of Google's ViT. CNNs are exceptionally good at extracting local features due to their convolutional layers, which apply filters across small receptive fields of the input image. This enables CNNs to effectively capture spatial hierarchies and local contextual information, such as edges, textures, and patterns, that are crucial for identifying and localizing objects within an image.

In contrast, the Vision Transformer (ViT) processes an image as a sequence of flattened patches, treating it more like a text sequence. This approach allows the ViT to focus on global relationships between different parts of the image, leveraging the self-attention mechanism of transformers. While this is powerful for understanding the overall context and relationships in an image, it might initially overlook the finer, local details that CNNs are inherently designed to capture.

Another difference between the DETR model and our model was the approach to handling the object detection pipeline, particularly in the prediction and loss calculation stages. DETR simplifies the object detection process by treating it as a direct set prediction problem. It uses a

transformer encoder-decoder architecture to predict a fixed number of bounding boxes along with their class labels in a single forward pass. This eliminates the need for components like region proposal networks (RPNs) or complex post-processing steps like non-maximum suppression (NMS), common in many traditional object detection models.

Furthermore, DETR introduces a unique loss function, combining a Hungarian matching loss with the standard classification and bounding box regression losses. This matching loss effectively enables a one-to-one correspondence between predicted and ground truth objects, ensuring that each prediction is uniquely matched with a real object. This approach is a significant shift from typical object detection models that rely on independently predicting object locations and then refining or filtering these predictions through additional mechanisms.

On the other hand, our model, based on the Vision Transformer (ViT), still follows a more traditional object detection framework. After the feature extraction with ViT, it uses a separate head (the Object Detection Head) for class prediction and bounding box regression. Post-processing steps, such as applying NMS, are required to handle overlapping predictions and refine the final set of detected objects. The loss computation in our model is more conventional, combining cross-entropy for classification and a form of regression loss (like Smooth L1) for bounding box accuracy.

In essence, while our model leverages the advanced feature extraction capabilities of ViT, it adheres to a more traditional object detection workflow. In contrast, DETR streamlines the process by integrating transformer architecture and a novel loss function, thereby redefining the object detection paradigm. This fundamental difference in the two approaches most likely had implications in the overall performance and accuracy of each model, with DETR performing far better than ours when it came to object classification.

We intend to explore alternative model architectures for better accuracy. We would like to explore techniques such as Feature Pyramid Network (FPN) which enable a better learning of image data for detection tasks. We also intend to try to understand model behaviours and accordingly tune the model for optimal performance.

7. Ongoing Work

Since we did find the DETR model to perform significantly well, especially when it came to object classification, we decided of its key architectural elements for our new model. This decision was grounded in the belief that integrating these aspects would bring about substantial improvements, particularly in areas where our previous model faced challenges.

One of the pivotal features we adopted from DETR is the Transformer decoder. In DETR, the decoder plays a crucial role in refining the output of the CNN backbone, enabling the model to focus on the global context of the image. By incorporating a Transformer decoder into our new model,

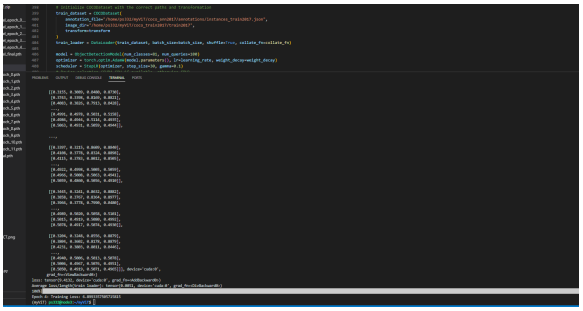

Figure 5. Epoch 4: Normalized Predicted Bounding Boxes and Loss Values for DETR implementation of Our Model

we aimed to enhance its ability to process and integrate comprehensive image features, leading to more accurate object detection and classification.

Alongside the Transformer decoder, we also implemented object queries. These queries, a distinctive feature of the DETR model, allow for a more structured and dynamic approach to object detection. Unlike traditional methods that often rely on predefined anchor boxes, object queries enable the model to learn to focus on specific parts of an image, making it adept at detecting a variable number of objects. This approach is particularly beneficial in complex scenes with multiple overlapping objects, where traditional models might struggle.

Another aspect where we drew inspiration from DETR is the design of the prediction heads. In our new model, the prediction heads for class labels and bounding boxes are applied after the Transformer decoder, allowing them to leverage the rich, contextual information processed by the decoder. This is a significant shift from the previous approach, where the prediction heads operated directly on the outputs of the ViT backbone. The new design enhances the model's ability to make more accurate and contextually relevant predictions.

Incorporating key aspects of the DETR model into our updated object detection framework, while adapting the loss function to our specific needs, has led to significant improvements. We drew inspiration from DETR's approach but customized the loss computation, focusing on classification and bounding box regression, without adopting the Hungarian matching algorithm. This tailored integration underscores our commitment to leveraging advanced technology to enhance our model's performance, particularly in complex scenarios. The results show promising advancements in detection and classification accuracy, and hopefully, we can have a model that is able to both classify objects with high accuracy as well as correct bounding box predictions. Due to time constraints and the complexity of the DETR model, as of right now, we are unable to provide mAP (mean Average Precision). However, we do have some starting numbers for training losses.

8. Conclusion

Our model demonstrates a valuable approach for leveraging pre-trained backbones and Transformers for efficient downstream computer vision tasks. Its simplicity and modularity offer advantages over existing methods, paving the way for further exploration in this promising direction. Although our model fell short of achieving desirable results, visual inspection reveals promising capabilities within the model. We think the insights for a simple yet modular model are valuable for future improvement of computer vision models.
[1]

References

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[4] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," 2020.

[5] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," 2021.

[6] Y. Li, H. Mao, R. Girshick, and K. He, "Exploring plain vision transformer backbones for object detection," 2022.

[7] B. Wu, C. Xu, X. Dai, A. Wan, P. Zhang, Z. Yan, M. Tomizuka, J. Gonzalez, K. Keutzer, and P. Vajda, "Visual transformers: Token-based image representation and processing for computer vision," 2020.

[8] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

1. Our code can be found on: https://github.com/pragyasharma123/ViTObjectDetection.git